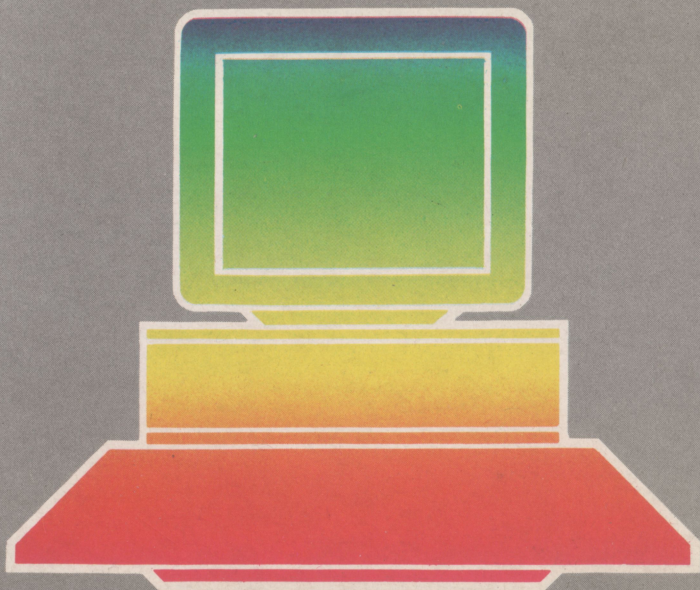
2

LE GUIDE DI BIT

UNIX

Lawrence Black Burn
Marcus Taylor

Lawrence Black Burn
Marcus Taylor

Prima edizione: 1984
Titolo originale: POCKET GUIDE UNIX
Editore originale: PITMAN PUBLISHING LTD

SUPERVISIONE TECNICA: Vittorio Riva
TRADUZIONE: Emilio Delle Piane
COPERTINA: Silvana Corbelli
FOTOCOMPOSIZIONE: Cencograf-Rotografica srl
P.zza S. Marco 1 - Milano - Tel. 655.20.13 - 655.51.45
STAMPA: Rotolito Lombarda S.p.A.

Indice

Come usare questa guida tascabile

Questa guida è stata presentata in otto sezioni per introdurre il sistema operativo UNIX ed il relativo software; può anche servire come veloce guida di riferimento per l'utente che sta ancora familiarizzando con i sistemi operativi. È stata progettata come un aiuto alle esercitazioni, con la maggior parte delle utility dei comandi UNIX messe in pratica singolarmente.
Ciascuna sezione comincia presentando il suo contenuto; il corpo di ciascuna sezione è diviso in presentazioni individuali di una caratteristica di UNIX, dove ciascuna presentazione è una descrizione di poco più di una pagina e finisce con una sessione pratica. La guida è progettata per essere montata come un cavalletto che può stare vicino alla tastiera di un terminale UNIX. Soltanto meditando effettivamente sui comandi UNIX nella sessione pratica e sottoponendoli praticamente ad un sistema UNIX otterrete la confidenza che vi porterà finalmente ad approfondire l'interfaccia comandi del sistema.
Non preoccupatevi di commettere errori! Nello scovare le correzioni e nel riprovare i comandi, acquisterete familiarità con la tastiera, ottenendo una maggiore conoscenza su cosa è e cosa non è consentito.
UNIX ha il vantaggio di essere standardizzato su diversi modelli di computer, grandi e piccoli: sta rapidamente diventando un potente strumento di sviluppo di sistemi per applicazioni commerciali su microcomputer a 16 bit. La descrizione di questa guida dovrebbe adattarsi al vostro sistema ma dovreste leggere il vostro Manuale di Programmazione UNIX e rilevare ogni differenza da questo standard mentre provate questi esercizi sulla vostra macchina.

Nota: UNIX è un marchio registrato dei Bell Laboratories Inc. ai quali l'editore non è in alcun modo legato.

1 Il sistema operativo UNIX

Un sistema operativo è un programma che gestisce le risorse di

un computer; esso si assume tutti i lavori di routine connessi alla gestione dello spazio sulla memoria di massa, al caricamento in memoria delle informazioni, al trattamento delle periferiche di comunicazione dati, e così via. Quando molti utenti sono collegati alla stessa risorsa del computer centrale, è il sistema operativo che arbitra le numerose e varie richieste che tutti gli utenti avanzano alle risorse del sistema.
Così come l'hardware è diventato negli anni più sofisticato e più vario, così hanno dovuto essere progettati sistemi operativi con maggiori capacità. I piccoli computer più semplici usano sistemi operativi elementari, di cui il tipo più semplice è un programma monitor che ha un insieme molto limitato di funzioni che di solito controllano un solo flusso di lavori attraverso il sistema. Sistemi più grandi e più potenti possono essere multiprogrammati, abilitando un numero di utenti ad ottenere simultaneamente un servizio dallo stesso sistema: questi sistemi necessitano di sistemi operativi più grandi e più complessi. UNIX, originariamente progettato per medi e grandi minicomputer, è ora disponibile su una grande varietà di macchine, spaziando dai super microcomputer (16 bit) fino ai grandi mainframe. Il sistema UNIX è moderatamente complesso e permette ad un numero di programmi di essere eseguiti simultaneamente sotto il suo controllo; ciò accade per mezzo dell'applicazione del principio del "time-sharing", che funziona dividendo ciascuna unità di tempo in un numero di porzioni. A ciascun programma in esecuzione (chiamato processo) viene data una porzione di tempo di esecuzione prima di passare al seguente processo in attesa. Ruotando i programmi in questo modo in ciascuna unità di tempo, tutti i processi avanzano: i computer moderni sono così veloci che possono essere soddisfacentemente serviti molti utenti.

Il kernel, i programmi ed i processi

Un sistema operativo come UNIX necessita di molti anni-uomo per il suo sviluppo ed è una grandissima collezione di moduli programma, più grande della maggiore parte delle memorie di computer. Poiché la memoria del computer è soprattutto usata

per i programmi e per i dati degli utenti, il sistema operativo è progettato in modo che soltanto le sue parti più frequentemente usate vengono conservate permanentemente in memoria, mentre la parte restante è mantenuta a portata di mano sul dispositivo di memoria di massa. La porzione di UNIX permanentemente residente in memoria è chiamata kernel (nucleo), e provvede alle esigenze immediate dei programmi utente in esecuzione ed assicura che si vada incontro a tutte le richieste degli utenti. La maggioranza delle funzioni UNIX non sono sempre necessarie e perciò vengono scritte come utility (o programmi standard) separate che sono chiamate dal disco su richiesta dell'utente: nuove utility possono essere progettate e aggiunte al patrimonio delle utility UNIX.
Un programma è una sequenza di istruzioni che devono essere eseguite dal computer. Prima che il programma venga eseguito deve essere caricato in memoria dal disco: se è in memoria ed in esecuzione all'interno di UNIX viene chiamato processo. Se un numero di utenti decide di usare il programma di utilità per copiare i file allo stesso momento, allora sarà generato un numero di processi ma vi è sempre una copia del programma (sul disco). Programmi ben progettati lavorano flessibilmente, cioè possono essere utilizzati in parecchie situazioni: ad esempio l'utility per copiare un file e dare al nuovo file-copia un nome qualsiasi. I programmi di utilità UNIX sono estremamente funzionali; questo significa che possono eseguire compiti specifici molto efficientemente. I tipici lavori dell'utente non possono perciò essere eseguiti da una sola utility UNIX; il lavoro dell'utente deve essere analizzato, spezzato in componenti funzionali, e poi deve essere trovata una utility UNIX corrispondente a ciascun componente. L'adattamento delle singole componenti nella giusta sequenza, normalmente produce i risultati desiderati.
In questo modo si è scoperto che UNIX è enormemente efficace nel risolvere un'ampia gamma di problemi di gestione delle informazioni degli utenti. Lo scopo di questa guida è di fornirvi sufficiente conoscenza delle comuni utility UNIX in modo che possiate identificare quali utility sono appropriate per risolvere i vostro problema.

Lo shell e l'editor

Lo shell (nome programa **sh)** è uno dei più importanti programmi del sistema UNIX; esso interagisce con l'utente che inserisce i comandi da interpretare ed appartiene ad una classe di programmi chiamati interpreti di comandi. A differenza di molti sistemi operativi il cui interprete di comandi è inserito nel kernel, lo shell UNIX è scritto come qualsiasi altro programma di utilità UNIX; comunque è un programma speciale poiché è sempre in uso per interpretare i vostri comandi. Proprio come un interprete, lo shell è anche un linguaggio di programmazione, e utenti UNIX esperti vorranno fare l'uso migliore di questa caratteristica dello shell.
L'editor (nome programma **ed),** cosi come lo shell, è interattivo nel suo uso e viene frequentemente usato dall'utente medio. Sebbene UNIX non abbia un editor standard, uno o due sono divenuti molto comuni, **ed** e **vi,** essendo **ed** un editor di linea convenzionale e **vi** un editor di schermo.
Entrambi forniscono all'utente un modo completamente flessibile di produrre e cambiare file di testo; per lo sviluppo di messaggi o programmi in codice sorgente e per la produzione di documentazione, l'editor di sistema gioca un ruolo centrale nella gestione delle informazioni dell'utente.

Comandi e argomenti

I programmi di utilità UNIX sono altamente funzionali ma per scopi generici. Questo è ottenuto con l'uso di argomenti che in generale seguono il nome dell'utility (o del comando).
Inoltre ciascun comando è progettato con un valore standard di ciascun argomento, cosicché i più frequenti modi di utilizzazione saranno per default (non è richiesto alcun argomento). Ad esempio, **is** senza alcun argomento produrrà un'elencazione minima di tutti i nomi di file nella directory di lavoro dell'utente, mentre **is/usr/bin** con l'argomento della directory espresso, produrrà un'elencazione simile ma per la directory nominata (/usr/bin). Parametri accoppiati con opzioni o flag modificano ulteriormente l'esecuzione del comando, offrendo una grande flessibilità agli

utenti. All'interno di questa guida molti comandi sono descritti con soltanto una parte della serie di opzioni, a causa delle limitazioni di spazio; gli utenti sono invitati ad esplorare i loro specifici manuali di programmazione UNIX per ulteriori dettagli. Nel testo ciascuna utility descritta ha un formato cioè: nome del comando seguito dagli argomenti. Alcuni degli argomenti sono opzioni e queste sono sempre introdotte, individualmente o a gruppi, con un trattino (-).

I processi in background

Una volta che il programma è stato caricato ed avviato, il processo può impiegare dei minuti o anche delle ore per terminare: questo impegnerebbe il vostro terminale per tutta la durata. Se volete continuare a fare qualcosa d'altro, UNIX ve lo permette, se vi prendete la pena di tramutare il lungo lavoro in un processo in background. Lo shell eseguirà il processo in background se aggiungete & alla linea comando.
Ad esempio se battete:

lavoro &

il programma **lavoro** viene caricato ed avviato come un processo in background prima di ripassare il controllo al terminale. Se il processo non interagisce con il terminale potrete ispezionare il suo procedere utilizzando il comando **ps** (stato del processo).

Output ed input standard

Nell'utilizzare sistemi di computer interattivi, il terminale è divenuto il dispositivo di comunicazione predominante: UNIX ha stabilito che lo schermo del terminale sarà l'output standard e che la tastiera del terminale sarà l'input standard. La maggior parte delle utility UNIX sono state programmate per elaborare informazioni provenienti dall'input standard o destinate all'output standard. Pertanto, la maggior parte delle utility possono essere facilmente unite in sequenza, con l'output di una che

fornisce l'input dell'altra, risolvendosi in un ulteriore elaborazione dei dati, sotto il controllo del programma shell.

Ridirezione dell'output

Poiché il programma shell controlla il flusso di dati tra il processo e l'input/output standard, gli si può comandare di ridirigere l'output da qualche altra parte, vale a dire ad un file o ad una periferica (considerata un file speciale). Ad esempio, **ls** lista normalmente la directory su un output standard (schermo VDU), **ls>miadir** metterà il listato nel file chiamato miadir. Il simbolo > dice allo shell di intercettare l'output e di ridirigerlo.

Ridirezione dell'input

Fino ad ora abbiamo incontrato il programma shell (l'interprete comandi) che accetta ingressi da input standard (tastiera). Lo shell può essere programmato per accettare comandi da un file su disco: ad esempio, **sh<mieicomandi** sottopone i comandi immagazzinati su disco nel file mieicomandi al programma shell (**sh**). In effetti un nuovo shell è ora in esecuzione sotto allo shell interattivo, mentre i comandi nel file sono elaborati: quando è finito, il controllo ritorna allo shell interattivo.

Pipe

Una pipe (:) connette l'output standard di un processo all'input standard di un altro. Poiché non sono coinvolti file intermedi questa capacità spesso risulta più elegante della ridirezione dell'input/output. Infatti in una serie di processi collegati insieme con delle pipe, tutti gli stadi intermedi di input/output sono invisibili all'utente (sebbene possono essere resi visibili); viene visto solo l'output finale. ALcuni comandi UNIX non ricevono alcun input standard (ad esempio **ls**) e perciò devono essere posti all'inizio di una sequenza di pipe; altri, come **lpr** (spooler di

stampa), non producono un output standard e perciò devono essere alla fine. L'uso delle pipe e la redirezione degli input/output sono le più potenti caratteristiche di UNIX, che offrono un grande potenziale per incorporare utility separate in sofisticati sottosistemi di elaborazione delle informazioni. Si raccomanda fortemente all'utente di familiarizzare con questi concetti utilizzandoli finché non siano conosciuti a fondo.

Il sistema di file UNIX

I file sono collezioni di informazioni dotate di un nome.
In UNIX sono riconosciuti quattro distinti tipi di file: testo, binario, directory e speciale. I file di testo contengono solamente informazioni codificate nel Codice Standard Americano per l'Interscambio di Informazioni (codice ASCII) e sono riconosciuti da quasi tutti i terminali e le stampanti. I file binari permettono di immagazzinare le informazioni utilizzando l'intero insieme dei codici disponibili, 256 per ciascun byte.
I file directory contengono liste di file, compresi altri file directory ove necessario, e ciò porta ad una importante caratteristica strutturale: il sistema di file UNIX è gerarchico. A ciascun utente è assegnata una directory chiamata home directory che è la posizione di partenza di accesso al sistema per quell'utente. Ciascun utente può andare e venire tra le directory ed anche creare o eliminare directory se la situazione lo richiede. Scoprirete che la natura gerarchica del sistema delle directory offre grandi possibilità per organizzarvi il vostro sistema informativo in modo da sapere esattamente dove e quando trovare ciò che cercate; tutti i file e le altre sub-directory sotto la vostra home directory sono sotto la vostra responsabilità.
Qualsiasi directory con la quale state lavorando diviene la directory corrente o directory di lavoro. UNIX dispone di un comando per dirvi in ogni momento quale è la vostra directory di lavoro **(pwd):** la directory di livello più alto nel sistema file è la directory radice (simbolo /) cosicché l'intero sistema informativo UNIX sta sotto questa in una struttura ad albero. Ogni volta che ci si deve muovere ad una directory di livello più basso per trovare un file si

pone come prefisso al nome del file un'altra barretta ed un nome di directory. I file speciali sono strani perché non contengono informazioni, ma servono solamente come canale adatto per periferiche che cercano di comunicare con il computer; se sono mandate loro informazioni, essi le passano all'appropriata periferica: ciascuna periferica possiede il proprio file speciale.

Tipi di file e modi

Oltre alle informazioni contenute nel file creato da voi o da qualche processo, UNIX ha bisogno di altre informazioni relative come il file deve essere usato ed a come è già stato usato. Le informazioni che descrivono il file includono i privilegi d'accesso, il tipo di file, date importanti per il file, dimensione del file e l'esatta locazione del file sul disco. Ad esempio, un file di testo non è un tipo di file che possa essere eseguito come un programma: UNIX lo sa in anticipo ed informa l'utente con un messaggio di errore se viene effettuato il tentativo.
È possibile vedere alcune di queste informazioni usando il comando **ls** con una delle sue tante opzioni, vi sarà fornito un listato di directory contenente praticamente tutti i dettagli dell'intestazione del file.
I dettagli del listato:
(Leggendo da sinistra a destra)
Modo file: un trattino significa non-directory; una 'd' significa directory
Privilegi del proprietario: (lettura scrittura esecuzione)
Privilegi di gruppo: (lettura scrittura esecuzione)
Privilegi di altri: (lettura scrittura esecuzione)
Numero di connessioni
Proprietario
Gruppo
Dimensione del file in byte
Dati di modifica
Nome del file
Il proprietario del file controlla l'accesso e l'uso del file modificando i vari privilegi con il comando **chmod.**

Questo conclude il nostro breve esame dei concetti base di UNIX. Tutti i comandi ai quali ci siamo riferiti e tutti i comandi comunemente trovati sono trattati ancora nelle sezioni seguenti. Tutto ciò di cui avete bisogno è un sistema UNIX su cui verificare le vostre idee.

2 I fondamenti del sistema UNIX

Ogni volta che ci si imbatte in una nuova situazione di studio, come l'iniziare un nuovo corso di insegnamento, dovrebbe sempre essere possibile trovare pochi principi basilari che permettano al principiante di cominciare la pratica fin dall'inizio. Questa sezione vi permetterà di sopravvivere al primo incontro con UNIX. È necessario assumere questo approccio perché UNIX ha un discreto grado di complessità strutturale che incorpora un certo numero di concetti riguardanti la manipolazione dei dati e delle periferiche su cui sono mantenuti. Comunque, se il lettore ha già familiarità con UNIX allora questa sezione può essere saltata.
UNIX è stato progettato affinché più utenti possano avere accesso "simultaneamente" alle risorse disponibili. Questo si ottiene per mezzo di algoritmi di "scheduling" (programmazione) che determinano e gestiscono situazioni casuali dove due o più utenti si contendono la stessa risorsa. I sistemi multi-utente necessitano anche di un accurato metodo per stimare quali risorse sono state utilizzate e da chi, così possono essere mandati i conti per il pagamento, se necessario. Per questa ragione, prima che sia possibile qualsiasi utilizzo, è richiesta una procedura di accesso al sistema, così come è richiesta una procedura di uscita. Queste procedure determinano con precisione gli indispensabili elementi usati nei processi di registrazione. Questa sezione riguarda le procedure di accesso e come le informazioni vengono organizzate nel sistema di file. Per permettervi di incominciare, in questa sezione sono considerate tre operazioni essenziali:

ingresso (login) identificazione utente ed inizio delle procedure di conto

owd stampa la working directory (directory di lavoro)

uscita	uso di **control-d** per terminare la vostra sessione interattiva

Questa sezione vi presenta anche la vostra primissima sessione pratica con UNIX. Ricordatevi che vedrete i vostri progressi se sarete in grado di sedervi ad un terminale collegato al sistema UNIX con la vostra Guida Tascabile accanto, cosicché la sessione pratica, che sta alla fine della maggior parte delle pagine, sia chiaramente visibile.

Ingresso nel sistema (login)—nome di conto e parole d'ordine (pass word)

Questa procedura permette ad UNIX di identificare gli utenti e predisporre i vari clock; inoltre fornisce all'utente un appropriato ambiente operativo.

Formato del comando - nome di conto e parola d'ordine

Qualsiasi combinazione minore o uguale a 15 caratteri alfanumerici può essere utilizzata per il nome di conto e per la "parola d'ordine".
Il nome di conto verrà riconosciuto da UNIX soltanto se è stato preventivamente predisposto dal system manager, il quale allo stesso tempo assegna i parametri ambientali, ad esempio l'appartenenza ad un gruppo di utenti, una home directory, una shell di login (tutti discussi in altre parti).

Esempi:
login:fred
password
fred viene battuto come nome di conto alla richiesta "login": del sistema, fred è riconosciuto da UNIX oppure respinto e seguito dalla richiesta ripetuta "login:". Se viene accettato e il nome "fred" è protetto da parola d'ordine, segue una richiesta di parola d'ordine. Vengono poi inseriti i caratteri della parola d'ordine, ma essi non appariranno sullo schermo per ragioni di sicurezza. Se la parola d'ordine corrisponde, i clock delle risorse cominciano a

funzionare per "fred", viene assegnata la home directory ed i comandi di login della shell vengon eseguiti prima di dare a "fred" il controllo sulla sua parte del sistema.

Pratica
Fate creare al vostro system manager un nome di conto per vostro uso: leggete le sezioni su **'passwd'** e **'nexpasswd'** per proteggere il vostro ambiente con la vostra parola d'ordine segreta. Leggete la sezione sull'uscita prima di accedere al sistema! Una volta entrati provate questi semplici comandi:

who (vi dice chi altro è su altri terminali)
date (vi dice quale data è caricata nei file di dati del sistema)
echo (stampa sulla linea seguente qualsiasi cosa lo segua)
esempio: echo ciao
Battete semplicemente questi comandi seguiti dal tasto RETURN o ENTER.

File e directory

File e directory sono concetti estremamente importanti nel sistema operativo UNIX; in effetti il programma UNIX è contenuto in una directory UNIX come file: un file è una collezione di informazioni dotata di nome. I file del computer, come i comuni archivi d'ufficio, hanno nomi, hanno una certa lunghezza, possono crescere, possono diminuire, possono essere creati o distrutti e possono essere letti. Prima che il programma venga eseguito, UNIX deve accedere al file, caricarlo in memoria e passargli il controllo: questo è un tipo di file eseguibile. Esistono altri tipi, ad esempio file testi che contengono soltanto caratteri ASCII ed i file directory che contengono informazioni (nomi dei possessori, date, lunghezze) su altri file e directory.
Gli utenti che lavorano con UNIX, tendono a creare e sviluppare file. Per permettere a molti utenti di fare ciò in modo concorrente senza intromettersi nel lavoro degli altri, tutti i file appartenenti ad un dato utente sono raccolti e mantenuti come un gruppo separato rispetto ad altri file. Questa raccolta è detta directory ed

esiste un file (tipo directory) per ciascuna directory che descrive in un record per ciascun file i contenuti della directory.
Quando il manager crea un conto utente, viene creata anche una home directory senza file; questa directory è la prima disponibile all'utente immediatamente dopo il completamento della procedura di login. All'interno di una directory l'utente può desiderare di archiviare all'interno di una sub-directory, una lista di file interrelati (vedere il comando ed). Gli utenti sono invitati a pensare ai modi di raggruppare logicamente i loro file e di creare directory in cui contenerli.
Un manager che scrive molti appunti per l'ufficio può costruire una struttura di directory all'interno della sua home directory per separarli tutti, ad esempio, per mesi, immagazzinando ciascun gruppo del mese sotto la corrispondente directory mese. Una struttura è costituita da file di appunti in una directory mese, e da directory mese all'interno probabilmente di una directory anno che, per contro, è in una home directory; questa è una struttura gerarchica ad albero ed UNIX permette all'utente di creare un numero qualunque di sotto-livelli. In questo esempio la lista delle directory nella directory anno è gen, feb... dic (tutti file directory); la lista nella directory feb è app1, app2 e app3 (tutti file di testo) e similmente nelle altre directory mese. Tipicamente una data directory contiene riferimenti sia a file directory che a file di testo.
Seguono semplici comandi di directory per illustrare questi punti.

pwd—stampa la directory di lavoro

Poiché per un utente è molto facile dimenticare quale directory sta usando, UNIX fornisce un comando che fornisce all'utente questa informazione; sapere quale directory si sta usando può essere particolarmente d'aiuto se il lavoro si svolge lungo la struttura ad albero.

Formato del comando
pwd non sono richiesti nè parametri nè opzioni

Esempio
owd se viene battuto in un momento qualsiasi, UNIX dovrebbe restituire qualcosa di simile a
/u/marcus/anno/luglio
Questo è il nome del file directory **luglio** contenuto nella directory **/u/marcus/anno;** se l'utente si è già spostato a questa directory (vedere il comando **cd),** allora il riferirsi all'appunto 5 è l'abbreviazione dell'intero nome di percorso di questo file, che è **/u/marcus/anno/luglio/app5.** Se il comando pwd dà la risposta **/u/marcus/anno** allora per riferirsi al file sopra dovrebbe occorrere **/luglio/app5.**
/u/marcus/anno/luglio/app5 è detto nome di percorso (pathname) completo del file **app5** poiché seguendo le directory in ordine da sinistra a destra porta l'utente lungo il cammino che alla fine conduce al file.

Pratica
Usate **cd/** per spostarvi in cima alla struttura ad albero; usate **owd** per vedere il risultato; usate **ls-1** per listare i contenuti della directory; prendete una nota di essi.
Domandate ad un utente esperto del sistema lo scopo delle directory nella vostra lista.

Uscita dal sistema (logout)—control-d

UNIX permette all'utente di uscire dal sistema molto facilmente: questo perché ciò libera un bel po' di spazio per altre persone. Battendo un carattere **control-d** al terminale (schiacciando il tasto control mentre si batte contemporaneamente d, senza preoccuparsi della directory in uso) si ferma la routine; poi le funzioni di accounting registrano gli estremi dell'utente e mandano il messaggio "login:". Se viene usato un collegamento via modem, si può riporre il ricevitore prima o dopo che **control-d** agisca.
Tra il login ed il logout, la maggior parte degli utenti comincia a pensare come utilizzeranno il sistema la prossima sessione; ciascuna sessione dovrebbe diventare sempre più produttiva ed

interessante. Sarà certamente così se si impiega un po' di tempo a scorrere il Manuale del Programmatore UNIX (MPU). Il MPU è abbastanza conciso ed in alcune parti difficile da capire per i principianti. Questa guida tenta di aiutare l'utente ad oltrepassare questo divario tra l'essere un completo novizio ed il sentirsi sicuri utilizzando la MPU.

3 L'Editor del sistema UNIX

Uno dei compiti più comuni che i computer effettuano è l'elaborazione di informazioni cosicché una volta rifinite siano più utili ed altre macchine o alle persone che le utilizzano. Prima che le informazioni possano essere elaborate devono essere inserite in qualche modo nel sistema. I computer acquisiscono informazioni da molte fonti: apparati sperimentali, casse continue, altri computer, tastiere dirette. L'ultima fonte, la tastiera, fornisce informazioni, attraverso un programma, ai file di dati contenuti sulle periferiche di immagazzinamento. La maggior parte degli utenti di sistemi UNIX lavorano nel campo della scrittura di rapporti, spedizione di lettere ed appunti, sviluppo di codici sorgente ecc., poiché il sistema operativo rende semplici questi compiti per mezzo di utility di supporto: **ed** è il programma che permette agli utenti UNIX di inserire informazioni dalla tastiera. Non vi è un editor standard per UNIX; in sistemi più recenti è stato inserito insieme all'**ed** un editor basato sullo schermo chiamato **vi.** Poiché **ed** è l'editor che si incontra più comunemente, qui sarà presentato brevemente quest'editor. L'editor è un programma interattivo: i comandi dell'editor sono usati per aggiungere del testo ad un file o per modificare il testo già nel file e, come l'interprete shell di UNIX, lo fa interattivamente. Il programma **ed** è in grado di soddisfare le vostre richieste per spostare testi, aggiungere testi, cancellare testi, includere testi da altri file, trovare e sostituire un testo con un altro: vi permette anche di dare una rapida occhiata ad un file. Approfondendo con successo **ed** aprite una finestra di comunicazione con i vostri colleghi. Unendolo con altre componenti di sistema (**pr** e **nroff**) esso include un potente package per il trattamento testi. Questa sezione è dedicata a

stabilire i fondamenti dell'utilizzo dell'editor del sistema UNIX.

File di testo

Il computer immagazzina le informazioni sul suo supporto di memoria in byte: ciò permette 2↑8 o 256 differenti valori binari. Per comunicazioni verbali generiche occorrono solo circa 100 valori differenti (uno per ogni A-Z, a-z, 0-9, ecc.), l'insieme di valori scelti a questo scopo è chiamato codice Standard Americano per l'Interscambio di Informazioni (ASCII): i file i cui byte contengono solo i codici ASCII sono chiamati file di testo.

Esempi di file di testo

Documenti
Tutte le informazioni usate per comunicare idee sono contenute in documenti come lettere, manuali, libri, appunti ecc. Se viene utilizzato un editor di testi su computer per creare questi documenti come file di testo, allora la loro futura manutenzione sarà semplice, utilizzando lo stesso editor.

Codice sorgente di programma
Al giorno d'oggi gli strumenti di sviluppo dei sistemi avanzati includono i compilatori, gli assembler, i data base manager, i generatori di rapporti, ecc. Le istruzioni di questi strumenti di sviluppo sono scritti in un linguaggio matematico o in uno simile all'inglese, contenuto in un file di testo. Il file di testo del computer contenente il codice sorgente può essere facilmente modificato utilizzando l'editor di sistema; questo è particolarmente utile durante la fase di sviluppo del programma sorgente.

Semplici file di dati
Indici, liste, glossari possono essere contenuti in un file di testo per essere ulteriormente elaborati da altri programmi (ad esempio programmi di ordinamento che trattano ogni riga come un record). Record più complessi che utilizzano molti campi nume-

rici possono essere immagazzinati più efficacemente usando un formato binario.

Editor di linea

Molti editor di testo sono chiamati editor di linea, questo perché l'unità base su cui essi operano è la linea di testo: ad esempio, cercheranno un file per trovare una data combinazione di caratteri e poi cancellarli, o per inserire dei caratteri o cambiare dei caratteri in quel punto. Il programma di editing "osserva" sempre una data linea nel file testo, indicata da un puntatore di linea che viene automaticamente incrementato o decrementato quando la linea esaminata cambia. Altri editor possono usare il carattere come loro unità di misura: ciò può essere più potente ma anche più scomodo da usare. L'approccio dell'editor di linea è stato ereditato dai tempi precedenti alle periferiche di visualizzazione ed alle basse tariffe di comunicazione, quando i terminali visualizzavano una linea per volta.

ed ed r — avvio dell'editor e lettura di un vecchio file

ed è un programma di supporto del sistema UNIX che parte battendo il suo nome quando UNIX è in attesa di un comando, una volta avuto accesso al sistema; il programma **ed** viene allora caricato ed attende vostri ulteriori comandi.

Formato del comando

ed (file 1)
Se un file di testo esiste già, allora l'utilizzo del nome opzionale del file dopo **ed** determinerà l'apertura del file (se presente nella vostra directory), e la lettura del buffer dell'editor, prima che il controllo ripassi all'utente.

Esempi

ed (senza nome del file)	per caricare l'editor di sistema pronto per l'uso

ed fred per effettuare operazioni di edit su un file
146 "fred" già esistente ("fred" è stato aperto con successo e 146 caratteri ASCII sono stati letti nel buffer di editing)

ed fred ("fred" non esisteva nella directory di lavoro indicato
? con ?)
ed parte l'editor
r memo5 ordina ad **ed** di leggere memo5 dalla directory
486 corrente: **ed** riferisce che ha letto 486 caratteri nel buffer di editing

Pratica
Vi è stato chiesto di preparare 8 rapporti su STATISTICHE DI VENDITE, uno a testa per otto differenti prodotti. Scegliete uno schema di denominazione di file ed attivate l'editor per creare il rapporto numero tre.

a, i, d, s — comandi base di editing: per aggiungere, inserire, cancellare linee e sostituire testi
Una volta attivato l'editor, esso può essere o in modo inserimento testi o in modo comando. Nel modo comando interpreta ed esegue comandi validi (ad esempio aggiunge ulteriore testo alla fine di un file). Nel modo di inserimento testi attende la linea o le linee da aggiungere o inserire (a seconda del comando); perciò finché siamo in questo modo, tutte le modifiche avvengono in un file buffer di memoria e non hanno realmente effetto finché il buffer non viene scritto come file.

Esempi
Per aggiungere linee di testo ad un file:
a (comando di aggiunta — append)
Index
a 1
b 20
c 49
(. pone fine al comando di aggiunta; aggiunte 4 linee)

Per spostare il puntatore di linea ad una data linea e scrivere la linea:
35p (pone il puntatore di linea a 35 e scrive)

Per inserire due linee di testo:
i (comando di inserimento)
questo testo viene inserito prima
della linea indicata dal puntatore
(. pone fine al comando di inserimento)

d cancella la linea di testo corrente

12d cancella la linea 12 del testo corrente

3.5d cancella le linee di testo da 3 a 5 incluse

s/vecchia/nuova/p sostituisce la "vecchia" parola con la parola "nuova" nella linea di testo corrente e stampa la linea corretta

Pratica
Create un file chiamato "messaggio1" ed aggiungete al file queste linee di testo (cominciando la Pratica); poi stampate la linea 1, la linea 2 ecc., usando il comando **p.**
Usate il comando **i** per inserire una linea di testo (va bene qualunque cosa) tra la linea 3 e la linea 4. In seguito utilizzate il comando **1,$p** (una variazione del comando **p**) per stampare tutte le linee del file per vedere se le vostre correzioni hanno funzionato; **$** immagazzina nel file il valore dell'ultimo numero di linea.

w,q — completamento dell'edit; scrittura del file, uscita
Dopo aver aggiunto, cancellato o modificato linee di testo, la

redazione del documento è completa ed i risultati devono essere salvati per usi futuri; il controllo viene poi ripassato al livello comandi di UNIX. A questo punto nel buffer di editing della memoria del computer esiste soltanto la nuova versione del documento: questo buffer deve ora essere scritto sulla memoria permanente del sistema di file.
Il comando **w** scrive il buffer di editing; il comando **q** ordina all'editor di finire e di ripassare il controllo all'interprete di comandi.

Formato del comando

w (nomefile)
q
Se l'argomento nomefile è omesso, viene preso il nome del file usato inizialmente nel comando ed, altrimenti l'editor avverte ed attende un nomefile. Se viene compiuto un tentativo di uscire prima che sia battuto il comando **w**, alcuni editor chiedono una verifica, nel caso in cui vi siate dimenticati di salvare i vostri raffinati sforzi!

Esempi

w appuntidioggi scrive i contenuti del buffer di editing nella directory corrente con il nome appuntidioggi
1466 informa che sono stati scritti 1466 caratteri
q l'utente esce dall'editor e torna al livello comandi di UNIX

Pratica

Usate l'editor per scrivere questa pagina nel buffer di editing utilizzando i comandi descritti negli ultimi tre paragrafi. Salvate la pagina come "edesempio"; uscite dall'editor e successivamente tornate ad editare "edesempio" usando **ed**, seguido da **r edesempio**. Leggete la pagina e riesprimete le frasi mentre la scorrete, sia cancellando intere linee che inserendone di nuove o sostituendo singole parole con altre.

4 Le Utility Unix

Uno dei principali punti di forza di UNIX è la sua abbondante disponibilità di programmi di utilità per risolvere problemi comunemente incontrati su una vasta gamma di applicazioni. Le utility possono essere classificate in tre insiemi: quelle che forniscono informazioni sull'ambiente del sistema in uso, quelle che riguardano la gestione di file binari e di testo (trattati nelle sezioni 5 e 6), quelle che sono disponibili al system manager. Questo insieme non è completamente sviluppato in questa guida, ma è presentato nella sezione 8. Entrando nel sistema UNIX, l'utente affronta una grande varietà di ambienti in cui lavorare; le utility elencate qui sotto permettono all'utente di accertare quale ambiente sta utilizzando.

pwd e **cd** directory corrente e cambio di directory
ls lista i file directory
file deduce il tipo di file
date, who data di sistema, chi è collegato
ps lista i processi
kill termina i processi
nohup esegue un programma mentre si è disinseriti
nice esegue processi a bassa priorità
time processi tempo
passwd cambia la parola d'ordine
find ceca un file
mail, write posta elttronica
stty, tty gestione terminale
du impiego del disco

Gli utenti troveranno **pwd** e **cd** in uso costante per ottimizzare i riferimenti ai file, richiesti dalla maggior parte dei comandi. I comandi **who, mail** e **write** permettono di predisporre servizi di posta elettronica tra gruppi in comunicazione. **du, time, ps** e **kill** permettono all'utente di valutare l'utilizzo del sistema e effettuare le azioni appropriate sui processi in esecuzione. L'utente che è in grado di coordinare questo insieme di utility con sicurezza, controlla l'ambiente di lavoro efficacemente e mantiene una buona sicurezza di dati sui file e sulle directory.

pwd—stampa la directory di lavoro

Sapere quale directory è in uso è di importanza cruciale per un giudizioso uso del sistema. L'intero sistema di file di UNIX è diviso in una moltitudine di sub-directory: un utente può essere in una di queste in un qualsiasi momento. Immediatamente dopo aver avuto accesso al sistema, l'utente è normalmente in una home-directory. Il comando **pwd** rivela in qualsiasi momento quale è la directory in uso; questo comando è frequentemente usato in combinazione con **ls** (per listare la directory) e **cd** (per cambiare directory).

Formato comando
pwd   nessun argomento o opzione

Esempio
Stampa il nome della directory di lavoro

pwd
/u/john   (il sistema stampa il nome della directory di lavoro)

Pratica
Accedete al sistema e trovate quale directory vi ha assegnato il vostro system manager (la vostra home-directory).
Prendete l'abitudine di verificare ogni tanto quale è la vostra directory corrente, particolarmente se il vostro lavoro vi ha richiesto di spostarvi tra la directory, usando il comando **cd**.
Ogni volta che cambiate directory entrate in una differente directory di lavoro ed il comando **pwd** stamperà un risultato differente.
Usate il comando **cd** (vedere il prossimo paragrafo) per passare alla directory radice (**cd/**); usate il comando **ls** (**ls-l**) per listare i file e le directory di quella directory e prendetene nota. Usate il comando **cd** per spostarvi a ciascuna directory ed il comando **pwd** per verificare quale directory è in funzione.

cd—cambio di directory di lavoro

Se utilizzando il comando **ls** per ispezionare un file appare una

lista inattesa, con **pwd** si verificherà quale è la directory di lavoro. Il rimedio all'essere nella directoy sbagliata è di passare alla directory giusta: **cd** fà proprio questo, su richiesta.

Formato comando

cd (nomedirectory)
Se "nomedirectory" non è presente, il sistema rende directory di lavoro la home-directory. Per risalire di una directory (la directory "padre" usate l'argomento .. e per passare alla radice del sistema di directory usate /.

Esempi

cd/usr/production Passa alla directory /usr/production
cd.. Passa alla directory "padre"
cd/ Passa alla directory radice

Pratica
Utilizzate **cd** per passare alla directory radice (/) ed usando **ls** e **cd** alternativamente esplorate i contenuti di tutte le directory di alto livello; prendete nota di qualunque interessante scoperta. Fate pratica utilizzando questo comando (**cd**) congiuntamente a **pwd,** finché non vi sentite abbastanza sicuri degli effetti prodotti da **cd**; **cd** effettua il cambiamento e **pwd** riferisce dove siete. Se UNIX fornisce un messaggio di errore, avrete probabilmente cercato di passare ad un file non-directory. Utilizzate il comando **ls-l** (o **ll** in certi sistemi — vedere prossimo paragrafo) per determinare quali file sono file directory e quali sono file non-directory: quelli che sono file directory nel listato cominciano con una "d" ed i file ordinari iniziano con un trattino (-).

ls—lista i contenuti delle directory

Questo comando fornisce informazioni sulle directory e perciò è una utility vitale per aiutare nella gestione appropriata dei file e delle directory un utente. **ls** possiede molte opzioni, troppo

numerose per essere completamente discusse in questa sede, esaminiamo comunque le opzioni essenziali.

Formato del comando.

ls(-ltasr) (nomedirectory)

Se non viene fornito un nome di directory, viene listata la directory di lavoro. Le opzioni più comuni sono:

- **-l** produce un formato lungo, un elemento per riga contenente i modi dei file, il proprietario, dimensione dati creati e nome in ordine alfabetico
- **-t** produce una lista in ordine cronologico
- **-a** lista tutti i file nella directory inclusi quelli che iniziano con un punto pieno (.)
- **-s** include nella lista la dimensione del file
- **-r** lista la directory in ordine inverso

Nel formato lungo (opzione **-l**) una tipica linea appare come:

drwxr-xr-l lb bus 2984 sep 30 17:53 chapter1

Il nome della directory può contenere un carattere jolly (*) per raggruppare le directory che hanno parti comuni nei loro nomi.

Esempio

ls lista tutti i file nella directory di lavoro

Ps/bin lista i file nella **/bin** directory

ls/u/us* lista tutti i file che iniziano (ad esempio) in **/u/us1**, **/u/us/2**, **/u/us3**

li-sr lista tutti gli inserimenti di directory in ordine inverso rispetto alle dimensioni.

Pratica

Utilizzando il comando **ls** e le sue opzioni ottenete le statistiche per tracciare un istogramma dei numeri di caratteri caricati nella vostra directory ciascun mese (o settimana) negli ultimi mesi (o settimane).

file—determina il tipo di file

Un listato di directory non fornisce alcuna informazione sull'origine e la natura del file listato. Il comando **file** informa l'utente se il file è un file di testo, directory o speciale e, se è un file programma, quale è il linguaggio sorgente.

Formato del comando

file (nomefile)

Esempi

file * lista tutti i tipi di file nella directory di lavoro
file /bin lista i tipi di tutti i file nella directory **/bin**
file dump lista il tipo di file di un file chiamato "dump"
date Visualizza la data e il tempo o regola la data ed il tempo

Formato del comando

date(yymmddhhmm)

Un utente può soltanto visualizzare la data pre-regolata: il manager può opzionalmente includere l'argomento per regolare la data.

Esempi

date visualizza il tempo
date 8406212230 pone la data al 21 giugno 1984, alle 22,30 (richiede il privilegio di manager)

who Lista tutte i nomi di conto degli utenti collegati
Talvolta bisogna sapere a) quanti utenti hanno accesso alle risorse del sistema, b) quanti sono e c) quale linea è in uso: il comando **who** dice all'utente tutto ciò.

Formato del comando

who
Il sistema risponde con il nome di accesso, il tempo di accesso ed il numero di terminale per ciascun utente collegato

ps—lista informazioni sullo stato del processo

Il comando, spesso usato dai programmatori e dagli amministratori di sistema, permette all'utente di esaminare quanto sta accadendo all'intero sistema dei processi o soltanto ad un processo. Qualunque processo fuori controllo può essere identificato utilizzando il comando **ps** ed eliminato, con l'autorità richiesta, usando l'identificatore di processo nel comando **kill**.

Formato del comando

ps (opzioni) (ident.processo)
Il comando **ps** lista il nome del terminale controllante, l'identificatore di processo, il tempo cumulativo di esecuzione ed una linea di comando analoga. Usate le eguenti opzioni:

-a lista tutti i processi attivi
-k esamina i dettagli **ps** nel file/usr/sys/core (dopo un incidente nel sistema)
-l lista in formato lungo (per il manager di sistema)

Se viene incluso l'identificatore di processo allora vengono listate solo informazioni da quel processo.

Esempi

ps lista i processi correnti per il terminale
ps-l lista i processi correnti per il terminale in formato lungo
ps-l 1234 lista dettagli sul processo corrente numero 1234

Pratica

Mandate un file di testo allo spooler della stampante (usate il comando **lpr** nomefile) per stampare ed emettete immediatamente un comando **ps-l** per vedere quali processi sono generati da **lpr**.

kill—elimina un processo in background

Occasionalmente i processi iniziati con un comando devono essere arrestati o abortiti. Il comando **kill** trasmette segnali per far terminare i processi coinvolti conosciuti. Il numero di identificazione di processo normalmente deve essere ottenuto utilizzando il comando **ps** (vedere paragrafo precedente).

Formato del comando

kill (-numero segnale) identificatore di processo
Solo il manager può eliminare processi di altri utenti.

Esempi
kill 1234   termina il processo 1234
kill -3 55   termina il processo 55 e forma un segnale di scarico 3
kill -9 222   termina il proceso degenerato 222 (segnale 9)
La maggior parte dei sistemi UNIX permette di mandare un gran numero di codici segnale; consultate il vostro Manuale del Programmatore UNIX.

Pratica

Inserite un comando che richieda molto tempo (ad esempio un sort). Esaminate la lista degli identificatori di processo usando **ps** e poi utilizzate **kill** per terminare il processo.

nohup—esegue programmi mentre non si è collegati

Un utente occasionalmente può desiderare di eseguire un programma che impiegerà un considerevole lasso di tempo per completarsi, tempo che sarebbe meglio utilizzare da qualche altra parte: **nohup** inizia un programma e gli permette di continuare anche se l'utente è uscito dal sistema.

Formato del comando

nohup (comando) (argomenti)
Qualsiasi comando UNIX che segua il nohup continuerà ad essere eseguito anche se l'utente esce dal collegamento. L'output standard del processo viene inviato ad un file "nohup.out", a meno che non venga specificata nel comando la ridirezione dell'output.

Esempi
Ordinate il file "telnos" in modo che sia possibile scollegarsi mentre ha luogo l'ordinamento; il "telnos" ordinato deve essere ritrovato in un file "sortnos".

nohup sh lavoro	esegue il comando (contenente il file **lavoro**) e non aspetta che finisca (il comando **sh** esegue tutti i comandi del file **lavoro**)

Pratica
Formate un listato ricorsivo (usate **ls-l**) di tutti i nomi di file del sistema di directory di UNIX e ordinateli in un rigoroso ordine alfabetico, prima di immagazzinare il risultato in un file **alphadir**; non attendete che finisca (usate **nohup**!). Scollegatevi e tornate più tardi ad ispezionare il file "alphadir", poi rimuovete il file (vedere il comando **rm**).

nice—esegue i programmi a bassa priorità

Qualsiasi programma il cui output non sia urgente ma che

potrebbe condizionare pesantemente le prestazioni del sistema, dovrebbe essere eseguito nelle "ore libere" o a bassa priorità. **nice** permette all'utente di eseguire tali comandi senza causare un grande scadimento delle prestazioni degli altri utenti.

Formato del comando
nice (-incremento) (comando) (argomento)
Il valore di incremento (1-19) determina l'ammontare di riduzione di priorità: si assume 10 se non specificato diversamente. Solamente un utente privilegiato può usare incrementi negativi per aumentare la priorità di un comando.

Esempi

nice -19 lavoro esegue "lavoro" con la più bassa priorità
nice --19 urgente esegue "urgente" alla priorità massima (superutente)
nice -19 ls -l Fsr<tuttifile produce una lista ricorsiva della directory Fsr nel file "tuttifile"

Pratica
Scegliete un programma che sapete che impiega alcuni secondi o più per finire; allenatevi ad eseguirlo sotto differenti livelli di priorità utilizzando **nice** e registrate i differenti tempi di esecuzione (leggete anche il comando **time** per aiutarvi in questo esercizio — paragrafo successivo)

time—cronometra un processo o un comando

Il comando **time** fornisce all'utente alcune statistiche riguardo al completamento di un processo o di un comando, il tempo di sistema ed il tempo totale trascorso. I rapporti di tempo variano notevolmente con il carico di sistema (numero di utenti).

Formato del comando

time (comando) (argomento)
La presenza di argomenti in un comando ne modifica conside-

rabilmente l'esecuzione e influenza il tempo impiegato per completarlo.

Esempi
time sorte fred cronometra il processo **sort**
time ls-l cronometra il processo **ls** (listato lungo)

Pratica
Applicate il comando **time** al comando **ls**, utilizzando differenti argomenti, per scoprire gli effetti che questi argomenti hanno sui tempi di esecuzione del comando.
Scegliete un periodo di carico minimo per il vostro sistema UNIX ed utilizzate il comando **time** per ciascun comando a turno, usando il numero minimo possibile di argomenti; per ogni rapporto ricevuto prendete nota dei tempi di ciascun comando. La comparazione dei tempi indica approssimativamente la relativa richiesta che i vari comandi fanno al sistema: ovviamente l'introduzione di un numero crescente di argomenti può notevolmente alterare i rapporti, come provato dalla vostra prima esperienza sul comando **ls**.

man—stampa le informazioni del manuale
UNIX possiede in linea un manuale di sistema progettato per aiutare l'utente alle prime armi: ciascun comando è documentato con formati ed esempi. Sui sistemi minori, a causa delle limitazioni di spazio dei dischi, i file di testo che contengono i dettagli del manuale possono essere eliminati.

Formato del comando

man (-opzione) (sezione) (titolo)
Il system manager può utilizzare questo comando per costruire un manuale per la distribuzione agli utenti. Le opzioni includono:
-t genera una versione troff per fotocompositrice
-r genera una versione nroff per l'output
-w stampa il pathname dell'ingresso del manuale, ma non l'ingresso in se stesso

Sono disponibili altre opzioni.

Esempi

man -w ls stampa il pathname del comando **ls**

man ls stampa le sezioni del manuale per il comando **ls**

Pratica

Da ora in poi ricordatevi di completare la comprensione del vostro sistema locale usando il comando **man** per produrre una copia tangibile di tutti i comandi che comincerete ad utilizzare in futuro. Ponete questa copia in un file personale.
Ad esempio il comando **man kill>>miomanuale** aggiunge le informazioni sul comando **kill** al file di testo **miomanuale**.

echo—ripete gli argomenti della linea comando

Gli utenti costruiscono routine adatte nei file dello shell: in queste possono mettere messaggi per indicare cosa sta accadendo. Il comando **echo** viene anche usato per visualizzare i valori attuali delle variabii dello shell e gli argomenti dei comandi.

Formato del comando

echo (argomenti) Gli argomenti possono essere messaggi o variabili dello shell.

Esempi

echo $user Stampa il nome di conto (variabile dello shell $user)

echo Funzione Completata Stampa sul terminale video il messaggio "Funzione completata".

Pratica

Esaminate uno qualsiasi dei vostri file comandi dello shell ed identificate i punti dove potreste mettere messaggi di spiegazione utilizzando **echo**. Usate **ed** per modificare i vostri file dello shell e poi listateli.

stty e tty—il driver del terminale
UNIX permette il collegamento all'interno dello stesso sistema di un'ampia gamma di tipi di terminali; ciascuna periferica deve avere un driver per trasformare le informazioni correttamente. Il comando **tty** stampa il nome di un file speciale utilizzato come input standard e **stty** viene utilizzato per controllare le numerose opzioni disponibili al terminale.

Formato del comando
tty stampa il nome del file speciale usato come input
stty stampa un sommario delle opzioni correnti
stty -tabs sostituisce i tab con spazi
stty 2400 pone a 2400 la velocità di emissione-ricezione della comunicazione

Esempi
La manomissione dei parametri dei terminali non è raccomandata: avete veramente bisogno di sapere molto sui terminali e la comunicazione per poter interpretare le numerose opzioni mostrate da **stty**. Se avete particolari necessità per il vostro terminale contattate il vostro system manager.

mail e write—comunicare con gli altri

UNIX fornisce un sistema di comunicazione e di posta elettronica. È utile essere in grado di mandare messaggi direttamente ad altri utenti già collegati o predisporre la consegna per la prossima volta che si collegheranno. Il comando **mail** è utilizzato per leggere la posta che vi è stata spedita o per spedire della posta ad altri individui o gruppi di utenti. Il comando **write** serve per mandare messaggi ad altri utenti collegati.

Formato del comando
mail (nome utente)
---messaggio---
Ctrl d
write (nome utente)

---messaggio--- Questo messaggio si inserirà di forza in ciò che il ricevente sta facendo.

Esempi

mail Elena

per favore porta la calcolatrice (manda questo messaggio all'utente Elena)

Ctrl d

mail Emilio Stefano < invito spedisce il contenuto di una lettera preventivamente composta (ed archiviata come "invito") agli utenti Emilio e Stefano.

Notate l'uso della ridirezione dell'input (<).

Pratica

Provate ad irrompere in una conversazione in linea utilizzando il comando **write** (usate **who** per scoprire chi è collegato). Inventate un protocollo per dire "passo" e "passo e chiudo"; scoprirete che se continuerete a farlo, interromperete il loro invio reciproco di corrispondenza.

du—utilizzo del disco

Gli utenti dovrebbero verificare spesso quanto spazio del disco è occupato dai loro file. Specialmente nei college, l'utilizzo del disco serve a calcolare il tasso settimanale al quale sono utilizzate le unità di risorse distribuite. **du** enumererà i blocchi della memoria a dischi utilizzati in ogni directory nel sotto-albero corrente o in qualsiasi directory specificata.

Formato del comando
du (nome del directory)

Esempi

du stampa un sommario dell'utilizzo del disco della directory corrente e delle sub-directory.

du /u/john/reports stampa un sommario dell'utilizzo del disco di **u/john/reports**

Pratica
L'uso regolare di **du** vi aiuterà a determinare i vostri riferimenti di utilizzo e a mantenere i vostri file ordinati. In un periodo di uso assiduo, perché non scrivere dei promemoria in una agenda da usare in determinati giorni finché non si è formata una buona abitudine? (vedere anche il comando **df**)

passwd—cambia la parola di accesso al sistema

passwd permette agli utenti di stabilire la riservatezza di un file assegnando una parola d'ordine associata con il loro nome di conto. Può anche essere utilizzato per cambiare periodicamente la parola d'ordine per mantenere la sicurezza.

Formato del comando

passwd
Il sistema richiede la parola d'ordine corrente, se stabilita; l'utente dovrebbe stabilire una nuova parola d'ordine di più di cinque caratteri, usando un combinazione di lettere maiuscole, di minuscole e di cifre. Durante l'inserimento l'**echo** è disattivato ed il sistema richiede una immediata ripetizione prima di accettare la nuova parola d'ordine.

Pratica
Se ve ne fosse la necessità, stabilite il vostro schema personale per generare parole d'ordine basato su qualche vostra formula personale ed acquisite l'abitudine di cambiarla ogni pochi giorni.

5 Gestione dei file di testo

Nella sezione 3 abbiamo visto come l'editor del sistema UNIX

(**ed**) vi permetta di creare file di testo. Nel loro lavoro, i professionisti hanno continuamente la necessità di creare, leggere, trasmettere e in generale manipolare informazioni. UNIX ha inserite dentro alla sua struttura di comandi molte caratteristiche che permettono la preparazione molto sofisticata di documenti: ad esempio, nella predisposizione per la fotocomposizione di una bozza di libri e rapporti. Le più avanzate caratteristiche di gestione dei file di testo non vengono discusse in questa guida.
Dopo la creazione di un file originale di testo, possono essere applicati al testo ulteriori processi, per rifinire maggiormente le informazioni contenute in esso. Questa sezione considera i comandi UNIX a vostra disposizione che vi permettono di ottenere quanto segue:

cat Congiunzione di due file (concatenazione)
pr Stampa di file (aggiunge titoli, impaginazione)
lpr Listati di file (sulla stampante del sistema)
diff Comparazione di file (verifica di differenze)
diff Riordino di file (ordina linee di testo)
grep Ricerca su file di maschere di testo
Ciascun comando possiede un insieme associato di opzioni che rende il comando estremamente flessibile, permettendo all'utente di ottenere un vasto insieme di funzioni di manipolazione dei testi. Questi comandi insieme costituiscono il package di word processing di UNIX ed è interessante notare che tutte queste caratteristiche esistevano in UNIX ben prima che i word processor elettronici divenissero affermati sui computer per ufficio.
L'uso appropriato dei comandi di questa sezione produrrà marcati miglioramenti alla vostra abilità di trattare testi.

cat—concatena e stampa i file

In un certo periodo di tempo possono essere stati creati numerosi file di testo in una directory o in più directory. Se ora dovesse servire raccogliere insieme tutti questi file separati (ad esempio i capitoli separati di un libro) e stamparli insieme in una operazio-

ne unica, **cat** li concatenerà nell'ordine specificato e li stamperà sull'output standard.

Formato del comando

cat (-u) file1 file2... (>filen)
L'output è verso il terminale per default, ma utilizzando la ridirezione dell'output può essere creato un nuovo file (filen), nella directory di lavoro, che conterrà i file concatenati. L'uso dell'opzione **-u** sblocca l'output al terminale o al file, ed ha l'effetto di cancellare le linee vuote alla fine di una pagina.

Esempi

cat memo1	stampa il file "memo1" sul vostro terminale
cat memo1 memo2 memo3 memo4	stampa i 4 file specificati sul vostro terminale
cat memo1 memo2 memo3 > memogiugno	concatena i 3 file specificati con "memogiugno" nella directory di lavoro

Pratica
Concatenando ripetutamente lo stesso file nella vostra directory scoprirete quanti file sono consentiti nella lista che **cat** gestirà: riferitelo ad altri utenti del sistema. Continuate a raddoppiare il numero finché si ferma, tornando poi indietro.
Quando lo avrete scoperto usate il comando con la ridirezione dell'output ad un file "fred", poi usate **cat** su questo file con l'opzione di sblocco e notate gli effetti.

pr—titola e formatta i file per la stampa

pr permette all'utente di introdurre testi nel sistema senza considerare come apparirà alla fine il formato; effettuerà la preparazione finale inserendo titoli, margini, numeri di pagina ed incolonnando il testo, se richiesto.

Formato del comando

pr (opzioni)file1 file2... (>filen)

Le opzioni seguenti possono essere utilizzate per mettere a punto il formato del o dei file. Il file o i file specificati vengono letti ed il file formattato viene presentato all'output standard: possono essere effettuate ridirezioni dell'output per salvare il risultato.
Le opzioni possono essere separate da spazi o raggruppate insieme.

-h utilizza l'argomento seguente nell'intestazione al posto del nome del file

-ln crea delle pagine lunghe n linee (valore di default = 66)

-m stampa tutti i file simultaneamente, un file per ciascuna colonna

-n crea un output su n colonne (default = colonna singola)

+n inizia l'output alla pagina n (default = pagina 1)

-t non crea un record di identificazione come titolo o coda

-wn utilizza una larghezza di pagina di n caratteri (default = 72) per l'output su più colonne

Esempi

pr capitolo1 stampa il capitolo1 in formato normale sul terminale

pr capitolo1>capi1.pr come sopra, eccetto il salvataggio del risultato nel file cap1.pr

pr -3t capitolo incolonna l'output su tre colonne, eliminando titoli e code

pr -2+5160m capitolo1 capitolo1 contemporaneamente duplica il capitolo1 con le intestazioni e le code (default) usando una lunghezza del modulo di 60 linee, iniziando la numerazione delle pagine dalla numero 5

Pratica
Create sul vostro editor di sistema un testo chiamato capitolo1, consistente in righe composte di lettere "a"; provate il comando **pr** usando gli esempi qui sopra. Riferite qualsiasi risultato inatteso al system manager.

lpr—listati di file testo sulla stampante di sistema

Spesso l'elaborazione finale di un file di testo consiste nel produrne una versione su carta per l'archiviazione o la distribuzione ad altri, ad esempio un manoscritto di un libro o un appunto. Praticamente tutti i sistemi che adottano il sistema operativo UNIX possiedono una stampante come mezzo per produrre copie su carta. Poiché è possibile che molti utenti richiedano contemporaneamente l'uso della stampante, **lpr** assicura che si formi una coda ordinata, nella sequenza "primo arrivato, primo servito". **lpr** cerca in continuazione qualunque lavoro di stampa nella coda di stampa: ogni nuovo lavoro di stampa in arrivo, mentre **lpr** è occupato, viene semplicemente aggiunto alla fine della coda. Alcune versioni di UNIX hanno aggiunto il comando **lpq**: questo vi permette di esaminare lo stato della coda, visualizzando sul vostro terminale la lista di lavori di stampa in attesa di essere serviti da **lpr**.

Formato del comando
lpr (opzioni) (file1...)
Le opzioni comuni sono:
-c copia il file immediatamente prima della stampa
-r elimina il file dopo che è stato stampato

Esempi
cd /u/libro
lpr - capitoli capitolo2 capitolo3 Stampa i file "capitolo1", "capitolo2" e "capitolo3" dalla directory /u/libro e cancella i file dopo la stampa.
lpr -cr ultimenotizie Stampa l'ultima copia di "ultimenotizie" proprio prima della stampa e successivamente rimuove i file.

I comandi collegati sono:
lpq esamina e visualizza i compiti nella coda di stampa (notate l'dentificatore di coda del vostro lavoro)
lprm toglie un lavoro dalla coda di stampa (usate l'identificatore di coda per specificare)

Pratica
Esaminate la vostra directory e accodate tutti i vostri file alla stampante utilizzando una volta **lpr**. Successivamente preparate un file testo per la stampa, usando il comando **pr** e collegate con una pipe questo risultato alla stampante.

diff—verifica delle differenze tra i file testi

Spesso vengono create in sequenza cronologica delle famiglie di file di testo; differenti versioni possono apparire identiche ad una ispezione casuale, ma hanno delle differenze. **diff** vi permetterà di determinare quali differenze esistono e perciò di ricordarvi, ad esempio, quale file è il più recente.

Formato del comando
diff (-efdh) file1 file2
L'output per l'utilizzo di **diff** indica in cosa differiscono i due file, usando un formato che indurrebbe l'editor di sistema di UNIX a ricreare il file2 dal file1 (opzione -e). Se desiderate che **diff** ignori le differenze nelle linee causate dagli spazi o dai tab, allora dovreste usare l'opzione -b. Dovreste consultare il manuale di riferimento del programmatore UNIX per ragguagli sugli effetti di altre opzioni.

Esempi

Diff -b capitolo1 capitolo1a	Confronta capitolo1 con capitolo1a, per le differenze, ignorando i caratteri vuoti e i tab.
diff -e capitolo1 capitolo1a	Comando di stampa dell'editor di sistema che ricrea il capitolo1a dal capitolo1.

Pratica
Create due file di testo **ex1** e **ex2** identici, usando **ed**; effettuate alcuni cambiamenti in ex2 usando **ed**. Esaminate i due file e provate a indovinare i messaggi di sistema se è stato usato il comando **diff -e ex1 ex2**; verificate il tutto inserendo nel sistema il comando. Esercitatevi molto, finché non raggiungerete la padronanza.

sort—riordina linee di testo; fonde file di testo preventivamente ordinati

Di tanto in tanto è richiesto l'ordinamento di informazioni in una sequenza desiderata, ad esempio per formare una lista in ordine alfabetico di nomi di società. Il programma di ordinamento ordina le linee del file in ingresso e poi scrive le linee rielaborate su qualunque tipo di output venga nominato, o per default al terminale video. Per default viene presa come chiave l'intera linea, ma possono essere specificate numerose posizioni di campi chiave.

Formato del comando
sort (opzioni) (+pos1(-pos2)...) (-o nomefile) (file1...)
+pos1 e -pos2 vengono usati per indicare le posizioni di linea per la chiave di ordinamento; -o inserisce il nome del file a cui deve essere mandato il risultato ordinato. Se viene fornito più di un nome di file, si assume che essi siano elencati in base alla stessa chiave e perciò saranno fusi insieme.
Le opzioni comuni sono:

-c verifica il file in ingresso; lo riordina solo se il file non è già stato ordinato.
-m fonde i file in ingresso.
-u in un insieme di linee uguali sopprime tutte le linee eccetto una

Esempi
sort -u list1 Visualizza sul terminale in ordine alfabetico tutte le sillabe uniche in una lista di parole.

sort -m -o tabsort tabella1 tabella2 tabella3 Salva nel file tabsort il file ordinato risultante dalla fusione di tabella1, tabella2 e tabella3.
sort -r +15-30 società Mostra un file società in ordine inverso, usando la chiave che inizia al carattere in posizione 15 sulla linea e che finisce al carattere prima della posizione 30.

Pratica
Create un file **dirlist** usando la ridirezione dell'output del vostro comando **ls** (ad esempio: **ls>dirlist**). Ora ordinate "dirlist" (**sort dirlist**) in ordine inverso usando come chiavi di ordinamento le posizioni di caratteri 3, 4 e 5; salvate l'output in **dirlisto**. Osservate **dirlisto** e controllate la sequenza.

grep—ricerca di stringhe nei file

Questo programma UNIX ricerca un testo di riferimento fornito dall'utente. Per mezzo delle sue opzioni può essere indotto a comportarsi in modi differenti; di norma listerà ciascuna linea (con i numeri di linea) contenente il testo di riferimento fornito. Quando applicato a più di un file indicherà quale file prima di stampare la linea, a meno che venga utilizzata l'opzione -h.

Formato del comando
grep (opzioni) espressione (file1...)
Le opzioni comunemente trovate sono:

- **-c** provoca un conteggio delle linee solamente (non le linee)
- **-h** toglie i nomi dei file dall'output
- **-l** lista i nomi dei file che contengono linee contenenti un riferimento di testo che corrisponde all'espressione
- **-v** lista le linee che non contengono il testo di riferimento
- **-n** enumera le linee in output

Esempi
grep UNIX capit* Stampa tutte le linee di un insieme di file

(capit*) contenenti il testo di riferimento "UNIX"

grep -c UNIX capit* Lista solamente il numero di linee dell'esempio precedente

grep -l Roberts Lista tutti i nomi di file della directory di lavoro che contengano qualsiasi riferimento a Roberts.

Pratica
Concatenate tutti i file di testo nella vostra directory di lavoro e poi cercate per tutti la presenza della parola "la", stampando ciascuna linea numerata. Utilizzando le opportunità delle pipe, ripetete quanto sopra ma prima listate l'output **grep** ordinato sulle posizioni di carattere 6-8 incluse: controllate e correggete finché non sarete soddisfatti.

Gestione dei file Unix

Dopo che il sistema operativo UNIX sarà stato in funzione per un po' di tempo, gli utenti avranno raccolto un certo numero di file all'interno delle loro e delle altre directory. Alcuni di questi file possono essere classificati e visti solo dai loro utenti; alcuni possono essere ancora in fase di sviluppo, altri possono essere solo residui di attività precedenti, ecc. Così come cresce il numero dei file, cresce anche la necessità di tenere in ordine l'ambiente, cioé le attività di manutenzione iniziano ad impegnare il tempo dell'utente, cosicché si può risparmiare sia spazio prezioso sui dischi, sia il tempo di ricerca dei file.
Il sistema operativo UNIX ha anticipato questa necessità fornendo una serie di utility per la gestione dei file cosicché ciascun utente sia in grado di spostare i file da un posto ad un altro, di copiare i file per un'altra destinazione, di togliere i file, di rinominare i file e di esaminare le directory. Ciascun utente ha responsabilità individuali di gestione sui file posseduti.
Questa sezione esamina e spiega l'uso di
rm cancella file
cp copia un file

ln unisce un file (fornisce un nomefile sinonimo)
mv sposta un file all'interno o tra le directory
ls lista i nomi dei file in una directory
chmod cambia il modo (autorizzazioni di accesso) di un file
chown cambia il proprietario del file
chgrp cambia il gruppo che possiede il file
mkdir crea una nuova directory vuota
rmdir elimina una directory vuota esistente
Presi come insieme, questi comandi offrono tutto ciò che è necessario per permettere a ciascun utente di mantenere il controllo sui file posseduti personalmente. Sfortunatamente ciò richiede veramente una disciplina personale che si può ottenere solamente con l'esercizio, al fine di ottenere un uso efficiente dello spazio di allocazione del disco. Alcuni sistemi fanno pagare all'utente in base allo spazio occupato sul disco: questo è un ottimo stimolo all'ordine.

rm—elimina i file

Ogni volta che un file binario o di testo diviene ridondante dovrebbe essere tolto per creare spazio ad altri file; **rm** permette all'utente di specificate quale o quali file necessitano di cancellazione. A seconda della situazione, UNIX chiederà conferma all'utente.

Formato del comando
rm (-fri)file1...
Se vengono usate delle opzioni queste hanno i seguenti effetti:
-f forza la rimozione. Non chiede conferma all'utente quando non è stato dato il permesso di scrivere.
-i opzione interattiva, chiede sempre conferma all'utente prima della rimozione. Una risposta "y" (yes) effettua la rimozione
-r rimozione ricorsiva (fate attenzione), rimuove un intero sotto-albero di file e poi la directory! Quando usato, l'argomento è il nome della directory.

Esempi

rm -i capitolo1 cancella interattivamente capitolo1; UNIX chiede conferma all'utente prima della rimozione

rm -f capitolo* (assumendo che non esistano altri capitoli) Rimuove capitolo1, capitolo2, capitolo3, capitolo4... "di forza", cioé senza chiedere conferma all'utente prima della rimozione

rm -ri u/libro Rimuove interattivamente tutti i file della directory u/libro.

Pratica

Usate il comando **ls** per vedere la vostra directory di lavoro dei file: esaminatela con cura per determinare quali file sono inutili. Utilizzate il comando **rm-i** per rimuovere ciascun file interattivamente; rispondete con "y" per ciascuna rimozione; verificate, usando **ls,** per confermare le vostre azioni quando avrete finito.

cp—copia un file

Talvolta è importante fare una copia di un file: ad esempio potreste desiderare di avere maggior sicurezza contro le cancellazioni accidentali o per continuare a lavorare su un nuovo file che inizia con il contenuto di un file già esistente, per salvare la duplicazione. **cp** permette di copiare all'interno della vostra directory di lavoro o da o verso un'altra directory.

Formato del comando

cp file1 file2 oppure

cp file1 file2 file3 ... directory

Nel primo formato file2 viene creato con esattamente le stesse specifiche di file2 (eccetto il nome), se il nome file2 non esiste ancora. Se file2 esiste già, viene rimpiazzato da una copia di file1 ma con il nodo del file e la proprietà ereditata dall'originale file2. Nel secondo formato il o i file vengono trasferiti alla directory indicata, che deve esistere, pur mantenendo i loro nomi di file originali.

Esempi

cp libro.indice In/sicurezza Crea una copia di sicurezza di **libro.indice** nella directory In/sicurezza.

cp agenda.gen agenda,feb copia di agenda.gen per l'editing come agenda.feb.

cp /n/sicurezza/*/f (crea i nomi-file **/f/u/sicurezza**) copia tutto il contenuto di In/sicurezza alla directory **/f**.

Pratica

Utilizzate **ls** ed esaminate la vostra attuale directory di lavoro. Usate **mkdir** per creare una subdirectory **/sicurezza** ed utilizzate il **cp** per copiare in **/sicurezza** tutti i file che volete proteggere; verificate il contenuto con **ls**.

ln—crea uno pseudonimo (o puntatore) per un file esistente

Occasionalmente potrà essere conveniente avere più di un nome per un file: un nome può indicare la famiglia al quale appartiene il file, mentre un altro indica i contenuti del file, ad esempio memo3 e datibudget denominanti entrambi lo stesso file. **ln** permette di associare al file degli pseudonimi.

Formato del comando

ln file1 (file2)

Quando usato, file2 diviene lo pseudonimo di file1; se viene listata la directory usando ls -i, sia file1 che file2 vengono mostrati con lo stesso i-nodo. Se file2 non viene usato allora viene creato uno pseudonimo nella directory di lavoro con lo stesso nome dell'ultimo componente del pathname riferito a file1 (collegamento attraverso le directory).

Esempi

ln list5 inventario Crea lo pseudonimo "inventario" per "list5"

ln /u/john/costi costi Crea uno pesudonimo "costi" per il file **/u/john/costi**

ln ../u/sicurezza/copiafin Crea lo pseudonimo "copiafin" nella directory di lavoro per il file "copiafin" in **/u/sicurezza**

Pratica

Utilizzate **ln** per associare un file contenuto in una lontana directory con il nome di un file contenuto nella vostra directory di lavoro; poi continuate ad esaminare il file (usate **ed**, **ls**, **mv** ecc.) riferendovi al vostro pseudonimo della directory di lavoro: facendo questo non dovete cambiare directory per evitare di utilizzare il riferimento all'intero pathname, pur continuando a lasciare l'accesso immediato alla vostra directory di lavoro.

mv - sposta un file all'interno o tra le directory; cambia il nome ad un file

Per ipotesi l'utente scopre, durante una operazione di "razionalizzazione", che le directory sono piene di file che sono stati posizionati erroneamente o accidentalmente, o semplicemente hanno bisogno di essere spostati in directory appena create. Il comando **mv** permette all'utente di gestire le directory di spostamento eseguita dei file in nuove posizioni.
Dopo una operazione di spostamento eseguita con successo, il nome di un file scompare da una directory e riappare in un'altra, o in qualche directory sotto un nome diverso (cioè viene rinominato).

Formato del comando

mv file1 file2

o

mv file...:directory

Se il file2 esiste già ed è protetto dalla scrittura, l'utente sarà interrogato per poter evitare la protezione; il comando è usato piuttosto spesso per cambiare il nome ad un file.

Esempi

mv/u/autore/indice/u/autore2 Sposta un file chiamato "**indice**" della directory **/u/autore1** a **/u/autore2**

my capitolo1 capitolo2 Sposta e rinomina come **capitolo1** un file chiamato **capitolo1**

Pratica

Utilizzate **mv** per ottenere un listato completo di tutti i vostri file; considerate l'intero insieme di nomi listati e progettate un miglior schema di denominazione al fine di riordinare i vostri file. Ora usate **mv** per ribattezzare ciascun file e per conformarvi al vostro migliorato schema di denominazione.

chmod—cambia il modo di accesso ai file

Vi sono tre modi di accedere ad un file: lettura, scrittura ed esecuzione. Vi sono anche tre categorie di utenti: possessori di file, gruppi possessori di file ed tutti gli altri utenti. Il modo del file è definito al momento della creazione (ad esempio per mezzo dell'editor, per spostamento o per copia), ma può rendersi necessaria una successiva modifica per mezzo di **chmod,** particolarmente se il file è stato copiato in un'altra directory per altri usi.

Formato del comando

chmod modo file1...

Il modo è descritto con tre parti: who, op, permission (chi, operazioni e permessi).

L'insieme dei simboli permessi è rappresentato di seguito.

chi	operazioni	permessi
u (utente) -	(toglie il permesso)	**r** (lettura)
q (gruppo) +	(aggiunge il permesso)	**w** (scrittura)
o (altri) =	(assegna il permesso)	**x** (esecuzione)
a (tutti)		**s** (assegna l'utente o il gruppo)

t (modo testo)
u (utenti presenti)
q (gruppi presenti)
o (altri presenti)

Soltanto l'utente (possessore) dei file ed il super-utente (manager) possono cambiare il modo file; i permessi da s ad o di norma sono gestiti dal sistema: i modi sono anche eguagliati a codici ottali che possono essere usati al posto dei codici precedenti (vedere la guida del sistema).

Esempi

chmod a+r notiziario rende "notiziario" leggibile a tutti (a è assunto per default, per cui può essere omesso)

chmod go-rw segreti rende "segreti" non leggibile e non scrivibile da altri se non dal possessore di "segreti".

chmod a-w rende impossibile a chiunque (compreso il possessore) di creare qualunque file nella directory corrente.

Pratica

Scegliete un qualunque file nella vostra directory (prima usate **ls-1** per ispezionarla) ed annotate la regolazione dei modi. Sperimentate **chmod** su questo file ed ispezionate di nuovo la regolazione dei modi per vedere gli effetti. Ora pensate attentamente e resettate i modi come prima, usando ancora **chmod.**

chown, chgrp—cambio del possessore di un file, cambio del gruppo file

Spesso un utente "eredita" dei file da altri utenti del sistema (perché reinventare la ruota se qualcuno lo ha già fatto). A questi file saranno inizialmente assegnati le particolarità della precedente proprietà, indicate in una operazione di **ls-1** (lista lunga). Il comando **chown** trasferisce i dettagli di proprietà al nome di

possessore dato, che viene confrontato con tutti i nomi validi di accesso al sistema, contenuti nel file di sistema **/ect/passwd.** Inoltre, se all'utente è assegnata un nuovo nome di conto per l'accesso, **chown** dovrebbe essere utile per riadottare tutti i file esistenti sotto il nuovo nome.

Formato del comando
chown nuovoutente file...
chgrp nuovogruppo file...

Esempi

chown laura valori	Passa la proprietà del file "valori" all'utente chiamato **laura**
chown marco *	Passa la proprietà di tutti file della directory di lavoro all'utente **marco.**
chown fisica fisi*	Passa la proprietà di tutti i file che iniziano con **fisi*** al gruppo chiamato **fisica**

Pratica
Utilizzate **ls-1** per esaminare tutti i file di tutte le directory sotto il vostro controllo e considerate ciascun file per un eventuale cambio di proprietà; usate il comando **chown** per cambiare la proprietà quando serve.

mkdir—crea una directory

Allo scopo di mantenere per un utente una struttura all'interno insieme di file che ricadono nello spazio di proprietà, possono essere create directory e sub-directory in cui raccogliere insiemi di file. Quando l'utente si sposta in una certa directory **(cd),** nuova directory di lavoro, i nomi dei file immediatamente disponibili per mezzo del nome semplice del file sono quelli della nuova directory: questa può includere solo pochi file ma tutti sono rilevanti per il compito in questione.

Formato del comando
mkdir nomedirectory

Può essere creato un qualunque numero di nomi di directory con l'uso di **mkdir. mkdir** crea anche nomi "." e ".." nelle directory: il "." è uno pseudonimo per il nome della directory, e ".." uno pseudonimo per la directory "genitore".
Quando si tolgono nomi di file dalle directory, questi nomi non vengono mai toccati.

Esempi
mkdir libri crea una directory sussidiaria chiamata "libri".
mkdir /u/john crea una directory in **/u** chiamata **john** funzionante da un'altra directory: è richiesto un permesso per scrivere nella directory **/u.**

Pratica
Verificate l'intero insieme dei nomi di file nella vostra directory e raggruppateli in insiemi logici. Fate una directory per ciascun insieme e poi utilizzate il comando mv su ciascun file per spostarlo nella sua directory designata; usate **ls** per verificare la vostra directory e notate l'effetto.

rmdir—cancella una directory vuota

Lo spostamento di nomi di file delle directory spesso le svuota del tutto (eccetto per lo pseudonimo "." e ".."): in questo caso la directory può essere in eccesso. Se non viene più usata è logico cancellarla, il che dà ad UNIX meno lavoro da fare.

Formato del comando
rmdir directory
I nomi di directory nella lista devono essere tutti vuoti; se non lo sono, UNIX replica con un messaggio adatto. L'utente deve avere il permesso di scrittura nella directory "genitore" per poter eseguire un tentativo che abbia successo.

Esempio
rm * cancella tutti i file della directory di lavoro
rmdir/u/laura/libri cancella la directory /u/laura/libri

rm/u/libro/capitol* cancella tutti i file della directory che iniziano con capitol
rmdir/u/libro cancella la directory /u/libro; essa deve avere tutti i file vuoti.

Pratica
Utilizzate mkdir per creare una sub-directory, poi usate **ed** o **cd** per creare dei file in questa directory. Esaminate la directory utilizzando **ll**; poi usando **rm,** ed infine **rmdir,** riportate il vostro sistema di file allo stato originale.

7 La programmazione dello Shell UNIX

La maggior parte degli utenti sono aiutati meglio da un sistema operativo che fornisce flessibilità nel programmare i suoi comandi e semplicità nell'eseguire un programma. Spesso un sistema operativo apparterrà ad una di queste classi ma non ad entrambi; comunque UNIX offre entrambe le capacità nello shell. Lo shell è probabilmente il più prezioso programma di utilità del sistema UNIX, sfortunatamente esso è spesso sottoutilizzato, a causa della scarsa documentazione o dell'incompleta comprensione da parte dell'utente.
Fino ad ora tutti i comandi illustrati richiamavano la potenza interpretativa del sistema shell, interattivamente, dove un singolo comando, talvolta in combinazione con una ridirezione nell'input/output (> o <), o con l'uso di pipe (:), otteneva tutto ciò di cui l'utente necessitava al momento.
Comunque queste tecniche mettono in pratica solo una piccola parte delle capacità dello shell.
In questa sezione esamineremo i modi più potenti in cui le procedure automatiche (non più interattive) possono essere programmate per l'interpretazione dello shell. Come in tutte le programmazioni, l'utente deve pensare al futuro e cercare di anticipare tutte le probabili necessità delle procedure (di solito è impossibile), cosicché quando il programma dello shell viene eseguito, egli deve controllare che gli eventi inaspettati non gli causino la perdita del controllo, che potrebbe generare la distruzione del sistema informativo dell'utente.

L'esperienza di programmazione ha insegnato ai progettisti di sistemi l'importanza di avere delle capacità di linguaggio per il controllo di sequenze ripetute, per la verifica di certe condizioni e per contenere temporaneamente dei dati invariabili. Lo shell di UNIX offre queste caratteristiche, cosicché tutta la necessaria sofisticazione richiesta per un controllo efficiente è disponibile in forma di routine per il programmatore dello shell.
Questa sezione illustrerà i seguenti comandi:
sh per eseguire un programma shell
uso delle variabili dello shell
condizionale semplice — l'uso dei test
condizionale if (se)
sostituzione di comandi
Le strutture **for** e **case** non saranno discusse qui, ma sono disponibili per la programmazione dello shell. Questi comandi possono essere programmati e immagazzinati in un file di comandi o programma shell. Per maggiori dettagli consultate il vostro manuale dei programmatori UNIX.

sh—sottopone linee di comandi all'interprete shell

Vi sono tre modi in cui possono essere presentate allo shell le linee comando da eseguire; il primo metodo usa una semplice ridirezione dell'input, il secondo utilizza il nome del file contenente le linee di comando come argomento del comando sh, ed il terzo sottopone il nome del file come un comando allo shell interattivo, dopo avere cambiato il modo del file allo stato di esecuzione.

Formato del comando
sh (<)nomefile o nomecomando
Il nome del file o il nome del comando sono file di testo all'interno di UNIX e sono interpretati dallo shell di sistema.

Esempi
Supponiate che ogni settimana sia richiesto un resoconto sulle directory opportunamente formattato e stampato in ordine di data di modificazione ed i comandi usati siano messi nel file

direp.week. Il file comandi potrebbe apparire così:
direp.>week
cd
date>directory.rpt
ls-lt>> directory.rpr
lor directory.rpt
echo Directory report è ora in coda di stampa — Ciao!
La procedura shell di questo file può essere richiamata in tre modi:
1 **sh <direp.week** oppure
2 **sh direp.week** oppure
3 **direp.week** dopo aver sottomesso **chmod a+x direp.week**
Pratica
Selezionate una serie di comandi semplici e separati; utilizzate **ed** per salvarli su un file chiamato **sh.test;** usate il comando **sh** in due modi (i modi 1 e 2 precedente) per verificare il vostro primo file di comandi. Verificate il vostro schema d'uso del sistema per trovare qualsiasi procedura normale che possa essere affidata ad un file di comandi, risparmiandovi così il tempo di battitura e ribattitura. Provate a convertire i vostri file, usando **chmod,** in file di comandi eseguibili dal sistema.

Variabili dello shell, assegnazioni di variabili, modi e valori

Un programmatore utilizza le variabili per immagazzinare valori che possono cambiare durante l'esecuzione del programma. I nomi di variabili dello shell UNIX devono cominciare con una lettera ma possono contenere lettere, cifre e sottolineature. Ad una variabile può essere assegnato un valore usando il comando di assegnazione ed il valore assegnato deve essere tra virgolette se nel valore sono richiesti spazi, tab o degli a-capo. Le variabili possono venire contrassegnate in modo che i loro valori non possono essere cambiati, con l'uso del comando **readonly** e se le variabili devono essere usate da altri file di comandi esse possono essere contraddistinte per il trasferimento per mezzo del comando **export.** I valori di una variabile possono essere visualizzati usando il comando **echo.**

Formato del comando
nomevariabile = valore
Il segno uguale è usato come simbolo di assegnazione
readonly (nomevariabile)
export (nomevariabile)
Usati senza gli argomenti, i comandi listano tutte le variabili di quella categoria.

Esempi
ty = Grazie
d₁= Domenica
area Z5 = Milano
readonly REG4
export REG4
echo $REG4 (occorre il segno di dollaro per stampare il valore di REG4; $ significa "valore di")

Pratica
Provate ad assegnare dei valori a dei nomi di variabili usando l'istruzione di assegnazione a verificate gli effetti dei comandi **readonly** e **echo** (ad esempio distinguete tra **echo pippo** ed **echo $pippo).** L'uso corretto delle assegnazioni di variabili è essenziale per costruire file di comandi complessi. Provate a riferirvi (interattivamente) ad una directory distante utilizzando il suo pathname completo come valore di una variabile, ad esempio **dir = /usr/a/b/c,** e poi usate **ls $dir** o **cat $dir/textfile.**

Argomenti del programma shell e variabili speciali dello shell

Le utility UNIX si affidano all'utente per poter fare un uso specifico delle loro capacità generali fornendo degli argomenti dopo il nome del comando; nei programmi dello shell gli argomenti delle linee di comando sono forniti al programmatore in una serie di variabili numerate; **$1** è il valore del primo argomento della linea di comando, **$2** è il secondo e cosi via. **$0** è il valore del nome comando in sè e **$#** è il valore del numero di argomenti usati nel comando.

Sono disponibili per il programmatore anche utili variabili dello shell che forniscono al programmatore (o ad un file dello shell) informazioni ad esempio sulla stringa del sentiero di ricerca (l'ordine delle directory in cui UNIX cerca il file) **$PATH;** il nome della home directory, **$HOME;** il separatore interno dei campi (di solito uno spazio), **$IFS;** le stringhe di prompt principali usate da UNIX, **$SP1** e **$SP2;** e il nome del terminale in uso, **$TERM.**

Esempi

echo $# stampa il numero di argomenti usati dall'ultimo comando sottoposto allo shell

if test $3>999 then echo 'argomento dovrebbe essere minore di 999'

if test $# = 4 (verifica 4 argomenti sottoposti)
then echo $4 $3 $2 $1 (li stampa in ordine inverso)
else echo Questo comando richiede quattro argomenti

echo $HOME stampa il nome della vostra directory sul terminale

PATH =:/bin:/usr:/u/mike/bin assegna la stringa del sentiero di ricerca per trovare i file

PSI = 'Fornire comando' cambia il normale prompt di sistema **$** in **Fornire comando**

Pratica

Controllate sulla vostra GPU i nomi di tutti gli argomenti dei programmi shell e le variabili shell speciali a voi disponibili. Eseguite un comando piuttosto complesso usando pochi argomenti e poi visualizzate interattivamente i valori di tutte le variabili e gli argomenti della vostra lista.

test, && e !! — operatori condizionale semplice e condizionale

Qualsiasi processo in esecuzione sotto UNIX termina normalmente (stato d'uscita 0) e o in modo anormale (stato d'uscita 1). Lo stato di uscita di un dato processo può essere usato per condizionare i processi che seguono logicamente o anche per abbandonare i processi seguenti. Il comando **test** comanda allo

shell di UNIX di esaminare lo stato d'uscita o dei processi e gli operatori condizionali **&&** o **!!** comandano allo shell di eseguire il processo seguente se lo stato d'uscita dal comando precedente è rispettivamente vero (zero) o falso (uno).

Formato del comando

test (argomento) (nomefile) (comando operatore condizionale)... dove l'operatore condizionale è **&&** o **!!** e l'argomento dipende dalla natura del test.

Esempi

test -d /u/pippo && echo Pippo ha una directory (le **&&** eseguono echo se vere)

test -d /u/pippo !! echo Pippo non ha una directory
(le !! eseguono echo se non vere)
(l'argomento -d verifica l'esistenza di una directory)

test $sconto -gt o && echo Conto è positivo (-gt rappresenta più grande di)

Pratica

Usando il vostro manuale del programmatore UNIX, esaminate quali altre condizioni possono essere testate sul vostro sistema. Sperimentate il comando test e i suoi operatori condizionali in modo interattivo finché non vi sentirete sicuri delle sue possibilità.

if—l'if condizionale

Gli operatori condizionali **&&** e **!!** sono utili nel caso delle liste. Lo shell offre l'**if** condizionale, tra gli altri, per aumentare la potenza di decisione.

Formato del comando

La sintesi dell'if condizionale è
if lista dei se
elif lista degli elif (**elif** = else if - "altrimenti se")

then lista degli allora
else lista degli altrimenti
fi (**fi** segna l'intervallo degli if condizionali)
Le parola **if,fi,elif** e **else** sono parole chiave riconosciute dall'interprete dello shell. Questa sintassi è la stessa che viene usata da molti dei più comuni linguaggi di programmazione.

Esempi
1 **if $risposta = si** (lista degli if)
then lpr mese,rpt (lista dei then)
else echo Rapporto eliminato (lista degli else)
fi
2 **if mattina then itervista avvisi**
elif pomeriggio then incontro pianificazione
else echo Verifica programi odierni
fi

Pratica
Scegliete una procedura amministrativa che potrebbe impiegare una serie di routine del computer. Analizzate la procedura nelle funzioni e determinate il procedimento logico in cui esse devono essere eseguite: nella vostra descrizione usate frasi condizionali semplici. Elaborate ulteriormente le frasi determinando le parti che sono equivalenti a variabili dello shell, a condizionali, a comandi ecc. e riscrivete la vostra istruzione originale usando le regole sintattiche di programmazione dello shell UNIX. Ora avete una istruzione strategica per lo sviluppo di applicazioni per questa procedura amministrativa: usate anche i comandi descritti nel paragrafo seguente.

While, until, do, done—cicli condizionali while e until

Talvolta un utente può aver bisogno di ripetere un gruppo di comandi che devono essere eseguiti tutti sotto certe condizioni. I condizionali while ed until permettono questa ripetizione finché prevale o mentre prevale una certa condizione.

Formato del comando

while lista-condiz.	**until** lista-condiz.
do lista-azioni	**do** lista-azioni
done done	**done**

Le parole **while, until, do** e **done** sono parole chiavi che delimitano l'area dei comandi da ripetersi. Mentre le condizioni della lista di while restano vere viene eseguita ripetutamente la lista di comandi dopo "do"; in modo simile, fintantoché le condizioni della lista until non divengono vere, viene ripetutamente eseguita la lista di comandi dopo "do"; un punto esclamativo dopo la lista while o la lista until negherà il senso della condizione.

Esempi

while test -r phonelist do sleep 300; done

Questa linea esamina ogni cinque minuti il file phonelist per vedere se il suo stato è leggibile (**-r**): lo shell non passerà al comando seguente nel file dello shell finché il file phonelist è leggibile.

while test ! -r phonelist do sleep 300; done

Questa riga è uguale alla precedente eccetto che per l'uso del punto esclamativo: lo shell non procederà finché non sarà leggibile il file phonelist.

until test -r phonelist; do sleep 300; done

Pratica
Combinte questi tre comandi con l'**if** condizionale per effettuare l'esercizio proposto nel paragrafo precedente.

8 Le utility Unix del manager

Sebbene questa guida sia destinata ai neofiti di UNIX, ci si aspetta che dopo un periodo di familiarizzazione essi diverranno abbastanza esperti da capire la necessità di una adeguata gestione e

controllo dell'ambiente degli utenti. Questa sezione tenta di dare una occhiata alle responsabilità che il manager di sisema si accolla ed ai comandi che sono utilizzati per mantenere l'integrità del sistema con beneficio di tutti gli utenti. Il manager di sistema è responsabile del mantenimento dell'efficienza globale e dell'affidabilità del sistema, installando nuovo software, copiando i file degli utenti, recuperando dati danneggiati o persi ed informando la comunità degli utenti dei nuovi sviluppi. Per i grandi sistemi UNIX queste responsabilità sono suddivise tra i membri di una squadra di supporto; nei piccoli sistemi, tutte queste funzioni sono assunte da una sola persona. A causa delle speciali necessità del manager di UNIX, viene dato il termine di super-utente al conto da lui posseduto; ai nomi di conto della radice e/o ai manager vengono concessi vari privilegi non disponibili agli altri utilizzabili solamente dai manager o dal conto di radice, che protegge il sistema dalla manomissione non autorizzata utilizzando i potenti comandi di sistema. I comandi più frequentemente usati per la gestione del sistema sono:

su diviene super-utente

mount, umount innesta e disinnesta una periferica

sync svuota i buffer di sistema

mknod crea file speciali

df stampa lo spazio libero sul disco

volcopy, cpio comandi di backup

fsck verifica del sistema di file

Incidenti di sistema e perdite di file riducono la fiducia degli utenti in qualsiasi sistema, a causa della mancanza di sicurezza: questi comandi servono per l'uso nei problemi di sicurezza. Il manager di sistema dovrebbe acquisire quanta più conoscenza possibile dei comandi del super-utente ed impiegare con cura questa conoscenza con riguardo alla sicurezza del sistema. Controllate il manuale dei programmatori UNIX per un elenco completo dei comandi del manager di sistema disponibili.

In questa sezione non sono comprese le sessioni pratiche poiché ad un principiante non sarà certo mai data la responsabilità di super-utente o manager, specialmente per far pratica di questi comandi.

su—passa allo stato super-utente

Per acquisire il privilegio di usare i comandi di questa sezione, l'utente deve avere lo stato di super-utente. Questo può essere ottenuto sia accedendo al sistema come radice che come manager, o emettendo il comando **su** dall'interno dell'ambiente di un normale conto; qualunque sia la strada usata, il tentativo sarà protetto per mezzo di una parola d'ordine. Se il tentativo ha successo, verrà emesso un diverso tipo di prompt di sistema prima di ciascun comando (di solito un #), per ricordare al manager o all'utente il tipo di stato. Se è stato emesso un comando **su** allora l'uso di **control-d** riporterà l'utente nello stato originale, altrimenti ricorre un logout, come al solito.

Formato del comando
su
Usate **control-d** per tornare alla posizione corrente nella directory dell'utente.

mount, unmount—innesta e disinnesta una periferica

I sistemi UNIX, come molti altri, hanno di solito un meccanismo per ricevere convenientemente software recentemente distribuito ed innestarlo all'interno della struttura dei file; per far ciò normalmente verrebbe usato un sistema a piccoli dischi (come un floppy disk), che è in grado di essere temporaneamente collegato al sistema mentre è in corso il trasferimento o su! quale viene effettuato l'accesso ai file all'interno del sistema di file collegato. Dopo l'uso, il sistema di file può essere scollegato con il comando **umount**, la periferica può essere resa disponibile per poter essere utilizzata da un altro sistema di file.

Formato del comando
mount nome directory di un file speciale (**-r**)
unmount file speciale
Il comando **mount** legge la directory del sistema di file trasportabile e la unisce alla directory del sistema di file UNIX, sotto il

nome di directory fornito per quell'uso. Una volta inserito, il sistema di file supplementare è accessibile nello stesso modo di tutte le altre parti del sistema di directory. Solitamente viene dato un avvertimento se viene effettuato il tentativo di rimuovere il volume prima di scollegare la periferica. L'opzione **-r** protegge il sistema di file innestato rendendolo solamente leggibile.

sync—svuota i buffer di sistema

UNIX mantiene continuamente delle tabelle di memoria vitali, particolarmente quelle concernenti il sistema dei file, che di tanto in tanto vengono scritte su disco. A queste tabelle si riferisce di nuovo il disco quando riparte il sistema. Pertanto dovrebbe essere dato un **sync** proprio prima di fermare il sistema, per indurre i buffer di sistema a scaricarsi sui dischi: aspettate finché l'attività del disco cessa, poi fermate senza pericolo l'attività del sistema.

Formato del comando
sync
Spesso i manager usano due volte il comando **sync**, per essere ben sicuri!

mknod—crea un file speciale

I file speciali sono un puntatore o un canale a dei dispositivi periferici che possono essere collegati al processore. Alcune periferiche sono guidate da una interfaccia a blocchi che trasmette i dati alla periferica (ad esempio disco o nastri) in blocchi di 512 byte; altre periferiche sono gestite da una interfaccia a caratteri che trasmette un carattere per volta alla periferica (ad esempio terminali, stampanti, ecc.). I file speciali vengono creati al momento della generazione del sistema o più tardi, quando viene introdotta una nuova configurazione: i file speciali possono essere trovati nella directory **/dev** (provate **ls -l/dev**). L'indicatore di modo del file può essere **b** (blocco) o **c** (carattere) e numeri di

periferica maggiori o minori indicano quale classe di periferica è stata definita, a seconda dei numeri principali.
Periferiche a carattere
0 console,tty1,tty2,tty3
1 tty
2 mem,kmem,null
3 rmt0
4 lp
5 rrk0,rrk1

Periferiche a blocchi
1 mt0
2 rk0,rk1

Formato del comando
mknod nome (**bc**) maggiore minore.

df—stampa un sommario dello spazio libero sul disco

UNIX richiederà spazio libero sul disco per mantenere affidabile la gestione dei file: circa il 20-30 per cento di spazio libero su disco. Il manager dovrebbe effettuare frequenti e periodici controlli su questa risorsa utilizzando il comando **df** ed intraprendere le azioni necessarie quando sono stati superati i limiti raccomandati. Tutti gli utenti possono usare questo comando ed ogni utente che intende costruire un file voluminoso dovrebbe usare **df** per verificare in anticipo che vi sia spazio sufficiente.

Formato del comando
df (nome directory)
Se non viene specificato alcun argomento viene fornito un rapporto su tutti i sistemi di file inseriti.

volcopy/labelit, cpio—comandi di backup

Il system manager deve sviluppare un procedura di backup di sistema per venir incontro alle necessità dell'installazione. UNIX

è d'aiuto fornendo varie utility che riflettono i differenti approcci alla copiatura dei dati. Qui vengono spiegati soltanto **volcopy/labelit** e **cpio**, ma altri sono **dump/restore**. **volcopy** copia interi sistemi di file da un luogo ad un altro, effettuando verifiche sulle label dei supporti (usando **labelit**) per assicurarsi che siano inseriti i volumi giusti per l'operazione di copiatura. Più tardi, se necessario, può essere ristabilito l'intero sistema di file o possono essere recuperati singoli file. **cpio** salva su nastro un sistema di file creando un grosso file contenente la copia completa del sistema di file. Anche questa copia può essere riletta selettivamente per ripristinare singoli file.

Formato del comando
volcopy
labelit
cpio
Verificate le dettagliate opzioni dei comandi nel vostro MPU.

fsck—verifica e ripara il sistema di file

Quando (non se) il sistema file diviene non attendibile è responsabilità del manager di sistema di riportare il sistema alla normalità. Le cause possibili sono numerose (fluttazioni di corrente, surriscaldamento dell'hardware, degenerazione delle superfici del disco ecc. La degenerazione potrebbe essere totale (coinvolgendo i backup) o parziale o con la perdita di un solo file. UNIX fornisce il comando **fsck** per aiutare il manager ad effettuare una efficiente verifica dell'intero sistema di file, se necessario. Esso riferisce dove esistono incoerenze e richiede azioni da parte del manager. In qualunque momento sia necessario il boot del sistema, **fsck** viene automaticamente messo in azione. Una operazione di riparazione **fsck** di solito significa eliminare le diramazioni inconsistenti e le foglie della struttura ad albero, cosicché alcune perdite sono inevitabili. Se si sospetta una generazione peggiore può essere più saggia una reintegrazione dell'intero sistema di file, sebbene i file utenti creati dall'ultimo backup andranno perduti. Talvolta i file risultano separati dai loro nomi (UNIX imma-

gazzina i nomi dei file in una directory ed il resto in un i-nodo (blocco di 64 byte con i dati del file. NdT)). I file che sono restati "orfani" vengono posti da **fsck** in una directory "lost+found" (persi e trovati) nella radice del sistema di file durante il processo di verifica; più tardi, ispezionando i contenuti di questa directory, potreste essere in grado di riconoscere il file e dotarlo di un nuovo nome, riconsegnandolo al proprietario originale.

Formato del comando

fsck (-y) (-n) (-sx) file speciale
fsck può essere applicato all'intero sistema dei file, ad una directory più bassa, o ad una directory inclusa.
Le opzioni significano
-y assume risposte positive "yes"
-n assume risposte negative "no"
-sx riscrive il superblocco ed ignora la lista libera

9 Sommario dei comandi e indice

Presentiamo qui di seguito un insieme abbreviato e ristretto dei comandi UNIX, per i riferimenti rapidi: la lista include più comandi di quelli inseriti nella parte principale della guida tascabile. Vengono usate le seguenti convenzioni per rappresentare i comandi:
Le parole in **neretto** devono essere batture così come sono.
Le parole da sostituire con stringhe specifiche (come il nome del file) sono sottolineate.
Le parole in caratteri normali sono di spiegazione.
Le parentesi attorno a un argomento indicano che questo è opzionale.
Il trattino (-) davanti ad un argomento indica una opzione; il trattino deve essere battuto se è richiesta quella opzione.

Le Guide di Bit sono uno strumento prezioso per chi lavora con il computer. In poche pagine riassumono con chiarezza quanto è necessario sapere su diversi argomenti di informatica, andando incontro alle più diverse esigenze.

Informatica
Forniscono le conoscenze fondamentali per una cultura informatica di base.

Software
Presentano i pacchetti software e i sistemi operativi più diffusi sul mercato e suggeriscono idee e proposte di programmi per diverse applicazioni.

Linguaggi
Sono comode tabelle di riferimento delle istruzioni e i comandi dei linguaggi più famosi.